**Affiliated To Anna University, Chennai & Approved by AICTE, New Delhi.**

**Coimbatore, Tamil Nadu, India – 641 021**

# 23MC104 – PYTHON PROGRAMMING LABORATORY

NAME :

BRANCH :

REGISTER NUMBER :

YEAR / SEMESTER :

ACADEMIC YEAR :

SUBJECT CODE :

SUBJECT NAME :

**Affiliated To Anna University, Chennai & Approved by AICTE, New Delhi.**

**Coimbatore, Tamil Nadu, India – 641 021**

# BONAFIDE CERTIFICATE

NAME :

ACADEMIC YEAR :

YEAR/SEMESTER :

BRANCH :

**UNIVERSITY REGISTER NUMBER:**

Certified that this is the Bonafide record of work done by the above student in the _____Laboratory during the year 2024-2025.

**Staff-in-Charge** **Head of the Department**

Submitted for the Practical Examination held on …….

**Internal Examiner**

# INDEX SHEET

| Sl. No. | DATE | EXPERIMENT NAME | PAGE NO. | MARKS | SIGN. |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| | |
|---|---|
| **Ex.no:1**<br>**Date:** | **Python Program to Count the Number of Vowels Present in a String using Sets** |

## AIM:

Python Program to Count the Number of Vowels Present in a String using Sets.

## ALGORITHM:

**STEP 1:** Start the program.

**STEP 2:** Initialize a set of vowels: vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}.

**STEP 3:** Prompt the user to enter a sentence and store it in sentence.

**STEP 4:** Split the sentence into a list of words using sentence.split(), and store it in words.

**STEP 5:** For each word in words, perform steps 6 to 11.

**STEP 6:** Initialize vowel_count to 0 for the current word.

**STEP 7:** For each char in word, perform steps 8 to 10.

**STEP 8:** Check if char is in the vowels set.

**STEP 9:** If char is a vowel, increment vowel_count by 1.

**STEP 10:** Continue to the next character in the word.

**STEP 11:** After counting vowels in the current word, print the word and vowel_count.

**STEP 12:** Continue to the next word in words.

**STEP 13:** After all words have been processed, output is complete.

**STEP 14:** Stop the process and exit.

**STEP 15:** End the program.

## SOURCE CODE:

```
# Count vowels without using a function
sentence = input("Enter a sentence: ")
vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}
words = sentence.split()
```

```
# Outer for loop to go through each word
for word in words:
    vowel_count = 0

    # Inner for loop to check each character in the word
    for char in word:
        if char in vowels:
            vowel_count += 1

    print(f"Word: '{word}', Vowels: {vowel_count}")
```

**OUTPUT:**

Output

```
Enter a sentence: pyhton
Word: 'pyhton', Vowels: 1

=== Code Execution Successful ===
```

**RESULT:**

| Ex.no:2<br>Date: | **Python Program to Remove the Given Key from a Dictionary** |
|---|---|

## AIM:

To write a Python program to remove the given key from a dictionary.

## ALGORITHM:

**Step 1:** Start the program.

**Step 2:** Define a dictionary with some key-value pairs.

**Step 3:** Display the original dictionary.

**Step 4:** Accept the key to be removed from the user.

**Step 5:** Check if the key exists in the dictionary:

- If the key exists, remove it using the pop() method.
- Otherwise, display a message indicating that the key is not present.

**Step 6:** Display the updated dictionary.

**Step 7:** Stop the program.

## SOURCE CODE:

```
my_dict = {
    'name': 'John',
    'age': 25,
    'city': 'New York',
    'profession': 'Engineer'
}

#Display the original dictionary
print("Original Dictionary:", my_dict)

#Input the key to be removed
key_to_remove = input("Enter the key to remove: ")

#Check if the key exists and remove it
if key_to_remove in my_dict:
    my_dict.pop(key_to_remove)
    print(f"Key '{key_to_remove}' removed successfully.")
else:
    print(f"Key '{key_to_remove}' not found in the dictionary.")
```

**#Display the updated dictionary**
print("Updated Dictionary:", my_dict)

## OUTPUT:

Output Clear

```
Original Dictionary: {'name': 'John', 'age': 25, 'city': 'New York', 'profession':
    'Engineer'}
Enter the key to remove: name
Key 'name' removed successfully.
Updated Dictionary: {'age': 25, 'city': 'New York', 'profession': 'Engineer'}

=== Code Execution Successful ===
```

## RESULT:

| | |
|---|---|
| **Ex.no:3**<br>**Date:** | **Even Number Pyramid Pattern** |

## AIM:

To write a Python program to display the even number pyramid pattern as shown below.

## ALGORITHM:

**STEP 1**: Start.
**STEP 2**: Input the number of rows for the pyramid from the user.
**STEP 3**: Initialize a variable even_number with the first even number (2).
**STEP 4**: Loop through the number of rows (i from 1 to n).
**STEP 5**: Print spaces to center-align the pyramid for the current row.
**STEP 6**: Loop through the range of i to generate and print even numbers for the current row.
**STEP 7**: Update even_number to the next even number after each iteration.
**STEP 8**: Move to the next line after each row is printed.
**STEP 9**: End the loop once all rows are printed.
**STEP 10**: Stop.

## SOURCE CODE:

```
# Program to print an Even Number Pyramid Pattern

# STEP 1: Get the number of rows from the user
rows = int(input("Enter the number of rows for the pyramid: "))

# STEP 2: Initialize the first even number
even_number = 2

# STEP 3: Generate the pyramid
for i in range(1, rows + 1):
    # Print spaces for center alignment
    print(" " * (rows - i), end="")

    # Print even numbers in the current row
    for j in range(i):
        print(even_number, end=" ")
        even_number += 2  # Update to the next even number

    # Move to the next line after each row
    print()
```

**OUTPUT:**

Output

```
Enter the number of rows for the pyramid: 5
    2
   4 6
  8 10 12
 14 16 18 20
22 24 26 28 30

=== Code Execution Successful ===
```

**RESULT:**

| | |
|---|---|
| **Ex.no:4**<br>**Date:** | **Write a program to find four different numbers using modules and packages** |

## AIM:

To write a Python program to find four different numbers using modules and packages.

## ALGORITHM:

**Step 1:** Start the program.

**Step 2:** Create a module (mynumbers.py) that contains functions to calculate the following four different numbers:

- Square of a number
- Cube of a number
- Factorial of a number
- Fibonacci sequence up to a given number

**Step 3:** Import the created module into the main program.

**Step 4:** In the main program, accept the required input values from the user.

**Step 5:** Call the respective functions from the module to compute the results.

**Step 6:** Display the output for each function.

**Step 7:** Stop the program.

## SOURCE CODE:

**File : mynumbers.py**

```
#Function to find the square of a number
def square(num):
    return num * num

# Function to find the cube of a number
def cube(num):
    return num * num * num
```

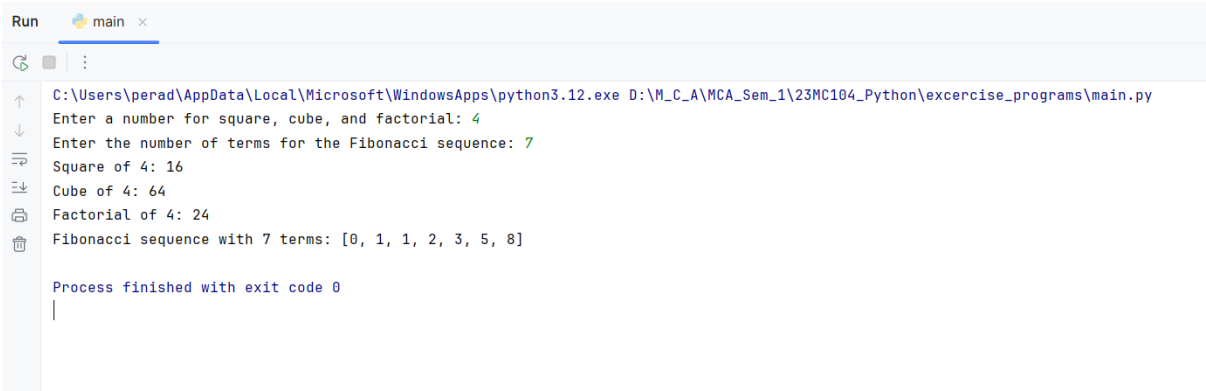
```
# Function to calculate the factorial of a number
def factorial(num):
    if num in (0, 1):
        return 1
    return num * factorial(num - 1)

# Function to generate Fibonacci sequence up to n terms
def fibonacci(n):
    fib_sequence = []
    a, b = 0, 1
    for _ in range(n):
        fib_sequence.append(a)
        a, b = b, a + b
    return fib_sequence
```

**File : main.py**

```
# Importing the module
import mynumbers

# Function to display results
def display_results():
    # Input from the user
    number = int(input("Enter a number for square, cube, and factorial: "))
    fib_terms = int(input("Enter the number of terms for the Fibonacci sequence: "))

    # Using the module's functions
    square_result = mynumbers.square(number)
    cube_result = mynumbers.cube(number)
    factorial_result = mynumbers.factorial(number)
    fibonacci_result = mynumbers.fibonacci(fib_terms)
```

```
    # Displaying the results
    print(f"Square of {number}: {square_result}")
    print(f"Cube of {number}: {cube_result}")
    print(f"Factorial of {number}: {factorial_result}")
    print(f"Fibonacci sequence with {fib_terms} terms: {fibonacci_result}")

# Main execution
if __name__ == "__main__":
    display_results()
```

## OUTPUT:




## RESULT:

| | |
|---|---|
| **Ex.no:5**<br>**Date:** | **Write a Python program to multiply all the numbers in a given list using lambda** |

## AIM:

To write a Python program to multiply all the numbers in a given list using a lambda function.

## ALGORITHM:

**Step 1:** Start the program.

**Step 2:** Define a list containing numbers.

**Step 3:** Use the reduce() function from the functools module in combination with a lambda function to multiply all the elements of the list.

**Step 4:** The lambda function will take two arguments and return their product.

**Step 5:** Display the result of the multiplication.

**Step 6:** Stop the program.

## SOURCE CODE:

```
#Importing reduce function
from functools import reduce

#Define the list of numbers
numbers = [1, 2, 3, 4, 5]

#Use reduce() with a lambda function to multiply all elements
result = reduce(lambda x, y: x * y, numbers)

#Display the result
print("The product of all numbers in the list:", result)
```

## OUTPUT:

```
Run    lambda ×

C:\Users\perad\AppData\Local\Microsoft\WindowsApps\python3.12.exe D:\M_C_A\MCA_Sem_1\23MC104_Python\excercise_programs\lambda.py
The product of all numbers in the list: 120

Process finished with exit code 0
```

## RESULT:

| | |
|---|---|
| **Ex.no:6**<br>**Date:** | **Write a program to create Circle, Rectangle, Square, Traingle, horizontal line using GUI** |

## AIM:

To write a Python program to create Circle, Rectangle, Square, Triangle, and a Horizontal Line using GUI.

## ALGORITHM:

**Step 1:** Start the program

**Step 2:** Import the tkinter library for creating the GUI.

**Step 3:** Create a main window using the Tk() class.

**Step 4:** Add a Canvas widget to the window where the shapes will be drawn.

**Step 5:** Use the canvas's drawing methods to create the following shapes:

- Circle: Use the create_oval() method.
- Rectangle: Use the create_rectangle() method.
- Square: Use the create_rectangle() method with equal width and height.
- Triangle: Use the create_polygon() method.
- Horizontal Line: Use the create_line() method.

**Step 6:** Display the window with all the shapes drawn.

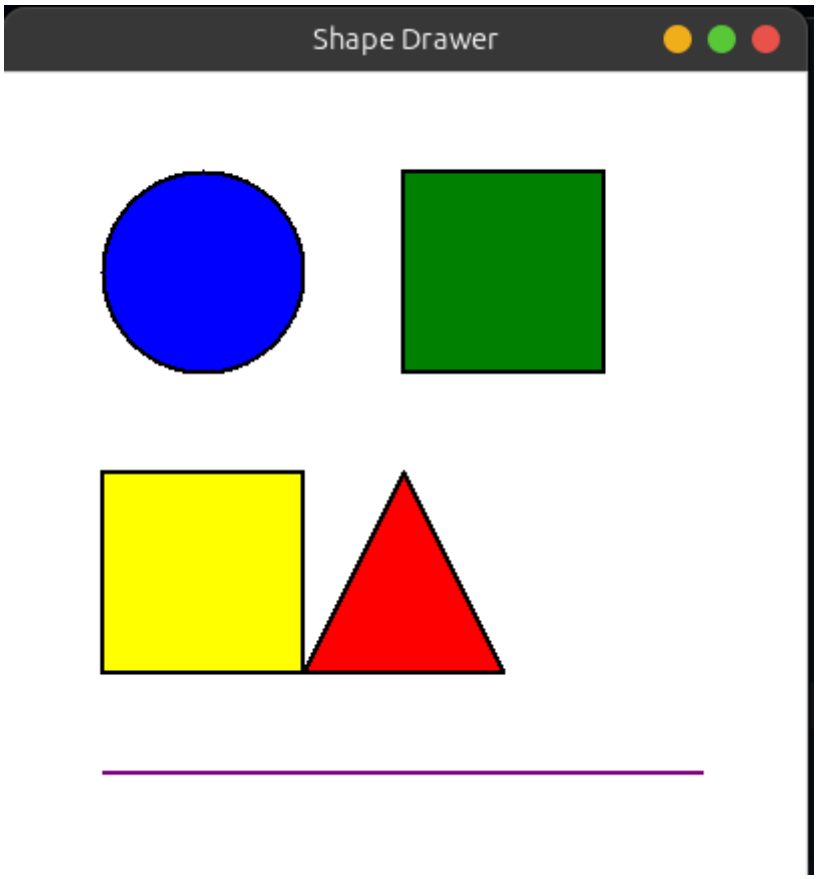
**Step 7:** Stop the program.

## SOURCE CODE:

```
import tkinter as tk
#Create the main window
window = tk.Tk()
window.title("Shape Drawer")
```

```
# Create a canvas
canvas = tk.Canvas(window, width=400, height=400, bg="white")
canvas.pack()

# Draw shapes

# Circle
canvas.create_oval(50, 50, 150, 150, fill="blue", outline="black", width=2)

# Rectangle
canvas.create_rectangle(200, 50, 300, 150, fill="green", outline="black", width=2)

# Square
canvas.create_rectangle(50, 200, 150, 300, fill="yellow", outline="black", width=2)

# Triangle
canvas.create_polygon(200, 200, 250, 300, 150, 300, fill="red", outline="black", width=2)

# Horizontal Line
canvas.create_line(50, 350, 350, 350, fill="purple", width=2)

# Run the main event loop
window.mainloop()
```

**OUTPUT:**




**RESULT:**

| | |
|---|---|
| **Ex.no:7**<br>**Date:** | **Python Program to Read a File and Capitalize the First Letter of Every Word in the File** |

## AIM:

To write a Python program to read a file and capitalize the first letter of every word in the file.

## ALGORITHM:

**Step 1:** Start the program.

**Step 2:** Open the file in read mode using the open() function.

**Step 3:** Read the content of the file using the read() method.

**Step 4:** Use the title() method to capitalize the first letter of every word in the content.

**Step 5:** Display the capitalized content.

**Step 6:** Close the file to release resources.

**Step 7:** Stop the program.

## SOURCE CODE:

```
# Open the file in read mode
file_name = "sample.txt"  # Replace with the path to your file
try:
    with open(file_name, 'r') as file:
        # Read the content of the file
        content = file.read()

        # Capitalize the first letter of every word
        capitalized_content = content.title()
```

```
        # Display the capitalized content
        print("Capitalized Content:\n")
        print(capitalized_content)
except FileNotFoundError:
    print(f"Error: The file '{file_name}' does not exist.")
```

## OUTPUT:


## RESULT:

| | |
|---|---|
| **Ex.no:8**<br>**Date:** | **Python Program to Read a Number n and Compute n+nn+nnn** |

## AIM:

To write a Python program to read a number n and compute n + nn + nnn.

## ALGORITHM:

**Step 1:** Start the program.

**Step 2:** Read the input number n as a string to easily form nn and nnn.

**Step 3:** Form nn by concatenating the string n twice.

**Step 4:** Form nnn by concatenating the string n three times.

**Step 5:** Convert n, nn, and nnn into integers and compute the sum:

result = n + nn + nnn.

**Step 6:** Display the result.

**Step 7:** Stop the program.

## SOURCE CODE:

```
# Read the input number
n = input("Enter a number (n): ")

# Form nn and nnn
nn = int(n * 2)  # Concatenate n twice
nnn = int(n * 3)  # Concatenate n thrice

# Compute the sum
result = int(n) + nn + nnn
```

**# Display the result**

print(f"The result of n + nn + nnn for n = {n} is: {result}")

## OUTPUT:


## RESULT:

| | |
|---|---|
| **Ex.no:9**<br>**Date:** | **Write a Python function that takes a list of words and return the longest word and the length of the longest one** |

## AIM:

To write a Python function that accepts a list of words and returns the longest word along with its length.

## ALGORITHM:

**STEP 1:** Start by defining a function to process the list of words.

**STEP 2:** Take a list of words as input.

**STEP 3:** Initialize variables:

- Set a variable `longest_word` to an empty string (`""`).
- Set a variable `max_length` to `0`.

**STEP 4:** Iterate through the list:

**STEP 5:** Find the length of the current word using the `len()` function.

**STEP 6:** Compare this length with `max_length`.

**STEP 7:** If the current word's length is greater than `max_length`:

- Update `max_length` to the current word's length.
- Update `longest_word` to the current word.

**STEP 8:** After completing the iteration, `longest_word` holds the longest word, and `max_length` holds its length.

**STEP 9:** Return the result as a tuple `(longest_word, max_length)`.

**STEP 10:** Print the returned longest word and its length.

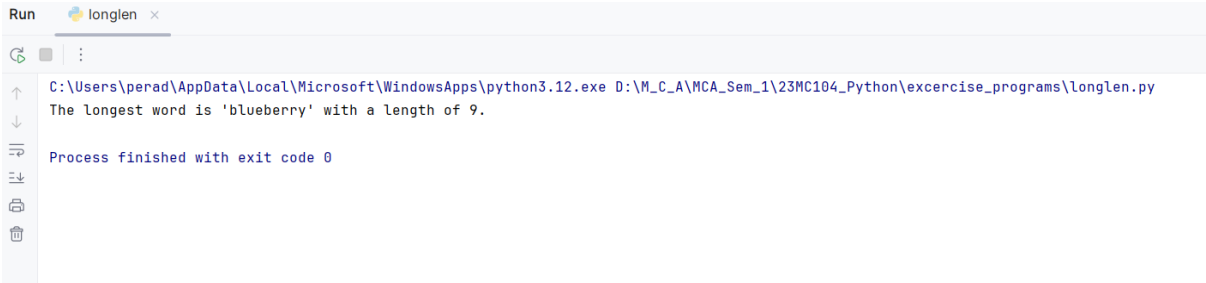
**STEP 11:** End the program.

## SOURCE CODE:

```
def find_longest_word(words):
    if not words:  # Check if the list is empty
        return None, 0
```

```
    longest_word = max(words, key=len)  # Find the word with the maximum length
    return longest_word, len(longest_word)
word_list = ["apple", "banana", "cherry", "blueberry"]
longest, length = find_longest_word(word_list)
print(f"The longest word is '{longest}' with a length of {length}.")
```

## OUTPUT:




## RESULT:

| | |
|---|---|
| **Ex.no:10**<br>**Date:** | **Write a Python program to count the occurrences of each word in a given sentence** |

## AIM:

To count how many times each word appears in a given sentence.

## ALGORITHM:

**STEP 1:** Start by taking a sentence as input.
**STEP 2:** Accept a string input from the user, representing the sentence.
**STEP 3:** Convert the sentence to lowercase to ensure the counting is case-insensitive.
**STEP 4:** Split the sentence into words using the `split()` method to create a list of words.
**STEP 5:** Create an empty dictionary to store words as keys and their occurrences as values.
**STEP 6:** Count the words:
**STEP 7:** If the word is already in the dictionary, increment its count.
**STEP 8:** Otherwise, add the word to the dictionary with a count of `1`.
**STEP 9:** Print each word along with its count.
**STEP 10:** End the execution of the program.

## SOURCE CODE:

```
# Accept the input sentence from the user

sentence = input("Enter a sentence: ")

# Split the sentence into words

words = sentence.split()

# Create an empty dictionary to store word counts

word_count = {}

# Count occurrences of each word
```

```
for word in words:
    word = word.lower()  # Convert to lowercase to make it case-insensitive
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1
```

**# Display the word count for each word**

```
print("\nWord Count:")
for word, count in word_count.items():
    print(f"{word}: {count}")
```

## OUTPUT:

```
Run    wordOccurence
C:\Users\perad\AppData\Local\Microsoft\WindowsApps\python3.12.exe D:\M_C_A\MCA_Sem_1\23MC104_Python\excercise_programs\wordOccurence.py
Enter a sentence: The quick brown fox jumps over the lazy dog while the curious cat watches from the nearby fence.

Word Count:
the: 4
quick: 1
brown: 1
fox: 1
jumps: 1
over: 1
lazy: 1
dog: 1
while: 1
curious: 1
cat: 1
watches: 1
from: 1
nearby: 1
fence.: 1

Process finished with exit code 0
```

## RESULT:

| | |
|---|---|
| **Ex.no:11**<br>**Date:** | **Write a Python Program to Compute the Area and the Perimeter of the Circle** |

## AIM:

To calculate the area and perimeter of a circle given its radius.

## ALGORITHM:

**STEP 1:** Begin the program by defining a function to compute the area and perimeter of a circle.

**STEP 2:** Accept the radius of the circle from the user.

**STEP 3:** Define constants - Use the value of π (pi) as `3.14159` or import it from Python's `math` module.

**STEP 4:** Calculate the area using the formula :

Area=π×radius2

**STEP 5:** Calculate the perimeter (circumference) using the formula:

Perimeter=2×π×radius

**STEP 6:** Display the calculated area and perimeter.

**STEP 7:** Stop the program.

## SOURCE CODE:

```
# Importing the math module for pi constant
import math

# Read the radius from the user
radius = float(input("Enter the radius of the circle: "))

# Compute the area of the circle
area = math.pi * (radius ** 2)

# Compute the perimeter (circumference) of the circle
perimeter = 2 * math.pi * radius
```

**# Display the results**

print(f"Area of the circle: {area:.2f}")

print(f"Perimeter (Circumference) of the circle: {perimeter:.2f}")

## OUTPUT:

```
Run    math
C:\Users\perad\AppData\Local\Microsoft\WindowsApps\python3.12.exe D:\M_C_A\MCA_Sem_1\23MC104_Python\excercise_programs\math.py
Enter the radius of the circle: 3
Area of the circle: 28.27
Perimeter (Circumference) of the circle: 18.85

Process finished with exit code 0
```

## RESULT: